# Stat 602 Project Report:

Kaggle's 'Don't Overfit! II'

*Kiegan Rice and Nate Garton*

*Due May 10, 2019*

# 1 Introduction

The goal of this competition was to try to train an effective binary classifier on a small training set with many predictor variables. The criteria used to determine the best solution was the area under the ROC curve generated with predictions on the test set. The training data set had dimensions $250 \times 300$, and the test data had dimensions $19750 \times 300$.

# 2 Exploratory Data Analysis

## 2.1 Data Quality

The quality of the data was pristine. There were no missing values in the test or training data. There was some class imbalance in the training set (roughly 64% of the observations were class 1). A 95%, two-sided confidence interval based on large sample normality assumptions for the population class proportion was $(0.58, 0.70)$. Thus, assuming that we were truly given a random sample from the total data, it seemed unlikely that the classes were truly balanced. However, even at the upper bound of the confidence interval, the class imbalance would not likely be large enough to suggest problems with typical classification models/algorithms.

There were 300 predictor columns given; however, there was no contextual information given about the predictors or the response. There was no information on whether columns represented any particular measured quantity, and each column was simply named $'0', '1', '2', \ldots, '299'$. This makes exploratory data analysis particularly difficult, especially since significant dimension reduction needs to occur to avoid overfitting to noise in the data, and since the training data set only has 250 cases. One clear approach in this scenario is to investigate distributions of each predictor variable, particularly class-conditional distributions since we are working with a classification problem.

## 2.2 Class Conditional Distributions

Because there are so many predictor variables, it is hard to pull useful information from tables of numerical summaries. Figure 1 presents histograms of four summary statistics: minimums, maximums, means, and standard deviations for each column conditional on the class. We can see that the ranges of all of the columns are similar between and within each class. We can also see that columns means for both classes are centered at zero and look fairly Gaussian. There may be one or two "outlier" means in class 0 that are unusually small, but given the number of columns in the data, it was not immediately clear to us that this wasn't simply an artifact of randomly sampled data. Standard deviations for both classes are centered at 1, and most standard deviations for both classes are between 0.9 and 1.1.

The behavior of these summary statistics seems consistent with data where most columns are $N(0, 1)$. With this in mind, we performed the Shapiro-Wilk test for normality on each column for each class. Assuming that the columns are independent for both classes, the distribution of p-values for each class should be approximately uniform. To test this, we performed Kolmogorov-Smirnov (KS) tests for the p-values from the Shapiro-Wilk tests. The two p-values from the (KS) tests were $\approx 0.904$ and $\approx 0.997$ for classes 0 and 1,

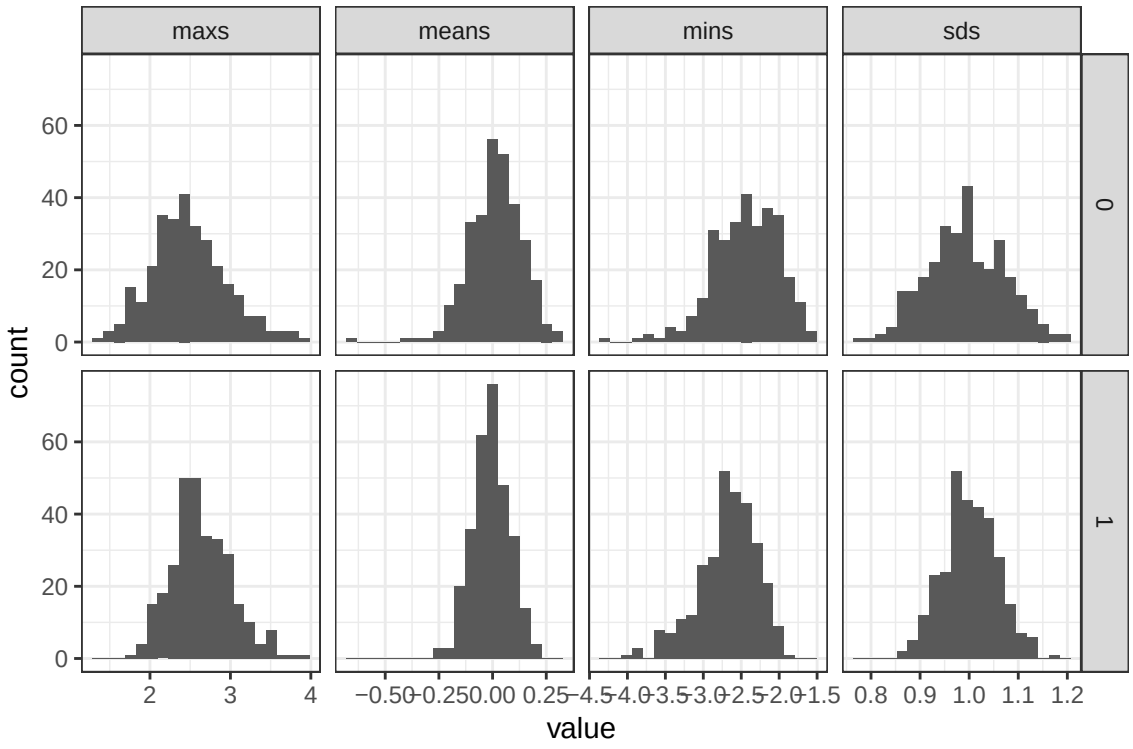


Figure 1: Class conditional histograms for column minimums, maximums, means, and standard deviations. Each row contains the column distribution for these four summary statistics for one of the two classes.

respectively. This led us to believe that assuming the normality of the predictor variables was likely reasonable.

If we could also conclude that the predictor variables within each class were independent, then this would allow us to effectively model the class conditional densities. To assess this, we first looked at histograms of the correlations. Figure 2 shows histograms of column correlations within each class. We see that both distributions are centered at essentially zero and appear to have a bell shape. Such a pattern is consistent with the sampling distribution of the correlation between two uncorrelated Gaussian random variables. The distribution of correlations for class 0 appears to be a bit more diffuse, but it is unclear exactly what to make of that. We performed z-tests after using the Fisher transformation for each correlation and then performed the KS test to test whether the p-value distributions for each class were close to uniform. The p-values from these tests were $\approx 0.113$ and $\approx 0.905$ for class 0 and class 1, respectively. We take these tests with a grain of salt because the correlations used in the test are not independent. However, if there were a handful of high correlations between several predictor variables, we might expect to see more p-values near 0. Finally, the fact that the correlation distribution for class 0 seems more diffuse than class 1 suggests that there might be some subtle differences in correlation structure between the two classes. However, performing a two-sample KS test comparing the correlation distributions between the classes results in a p-value of $\approx 0.251$. Again, the assumption of independence is violated, so we take this with a grain of salt. But, this serves to reinforce the idea that any difference in correlation structure between the two classes is subtle, if it exists. It is also worth commenting that while it is not true that marginally Gaussian, uncorrelated random variables are necessarily independent, if the true joint distribution is not multivariate normal, modeling class conditional

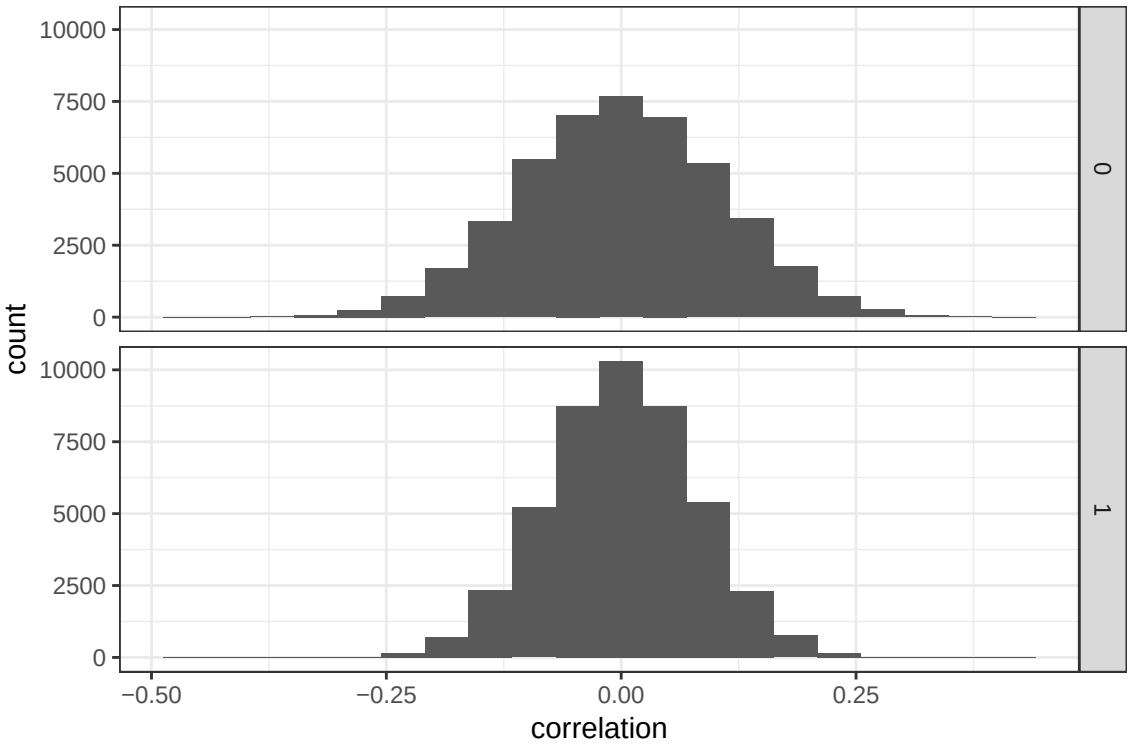


Figure 2: Histograms of column correlations for each class.

densities is a hopelessly challenging task.

Visually investigating class-conditional densities for each column also gives a good look at columns where there may be some slight separation between the two response classes, seen in Figure 3. However, additional statistical tests should be done to assess whether the densities are significantly different, especially since we have such a large number of columns. Some visual separation could be an artifact of random sampling for the training set.

We performed two sample t-tests of the difference in means (assuming equal variance) between each of the corresponding columns from the two classes. If we assume that the columns are all independent (which may be reasonable), then the Bonferroni correction appropriately controls type-I error without being overly conservative. Two columns have p-values that are significant at the corrected 0.05 type-I error level. Those variables are 33 and 65. Thus, it seems that the separation observed in those two distributions is unlikely to have resulted from sampling variability.

# 3 Feature Engineering

Creative feature engineering proved particularly difficult for this problem, especially due to the murky nature of the data. While data were pristine in the sense of completeness, the lack of contextual information about predictors makes context-based feature engineering impossible. With that being said, our exploratory data analysis suggests that the class conditional distributions can be modeled well under strong assumptions. In

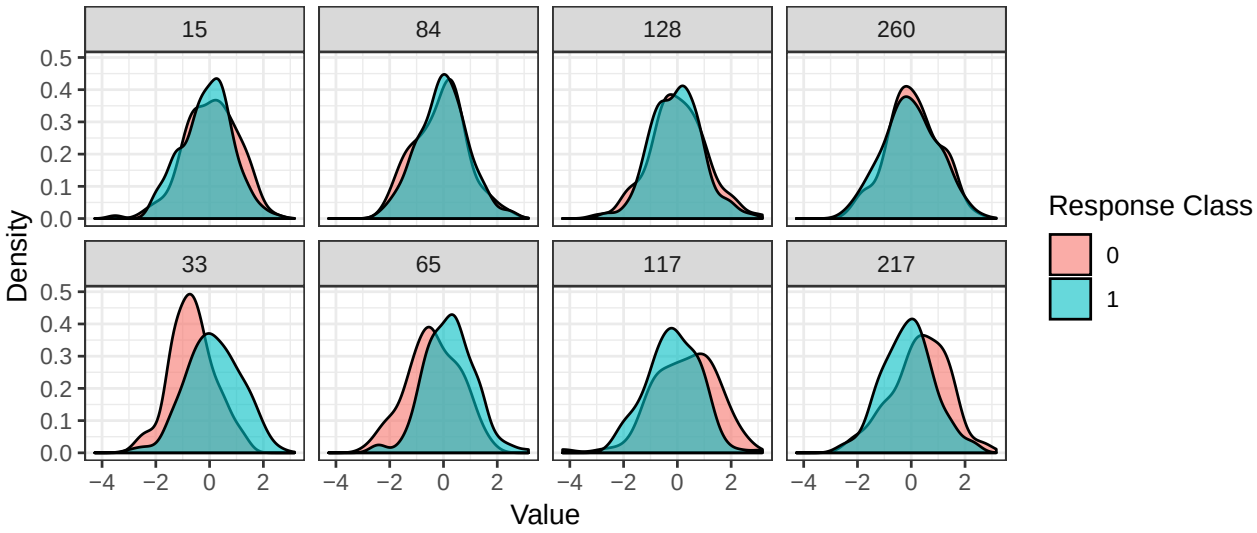


Figure 3: Example of class-conditional densities for several columns in the dataset. The first row shows class-conditional densities that do not demonstrate any separation, which is very common in the dataset. The bottom row demonstrates class-conditional densities observed in the training data which seem to show separation.

the absence of abundant data, it is critical to take advantage of reasonable assumptions. The large number of predictors also poses the challenge of weeding out noise without removing useful information; dimension reduction plays the first big role in constructing a good set of features.

We have already discussed how two predictor variables appear to be statistically significantly different across classes. However, we would rather do variable selection by optimizing for the criteria upon which we will be judged (AUROC) than using t-test p-values as the selection criteria. Therefore, rather than exclusively using the two variables that have the smallest p-values in our models, we opt for a more traditional type of machine learning variable selection technique. LASSO is a natural choice for dimension reduction here, as it will shrink many predictor parameter estimates to zero in favor of those that more adequately describe the response variable. A logistic regression LASSO model using all 300 predictors was fit using the `glmnet` package, in particular, the `cv.glmnet` function with $\alpha = 1$. Columns with non-zero parameter values in the $\lambda_{1se}$ model were kept for future feature engineering, while the others were removed from consideration.

Recall that we have argued that it is tentatively reasonable to assume that, conditional on the class, the predictor variable distributions are independent normal random variables with standard deviation 1. In this case, we should not need to worry about interactions between variables being important for classification, while differences in column distributions corresponding to mean shifts should be identified by the LASSO.

### 3.1 Likelihood Ratios as Features

Our team considered two different approaches to using likelihood ratios as features: an assumed normal distribution approach, and a kernel density estimate approach.

### 3.1.1 Gaussian Likelihood Ratios

Given the assumption that the class conditional distributions of the data are independent Gaussian with standard deviation 1, then the optimal feature is the product of univariate Gaussian likelihood ratios with different means. We estimate the means (conditional on the class) for each of the variables retained by the LASSO using $\lambda_{1se}$. Even though it seems reasonable to assume standard deviation 1, we estimate the standard deviations for each Gaussian likelihood as well. Let $f(x|\hat{\mu}_{0i}, \hat{\sigma}_{0i})$ and $f(x|\hat{\mu}_{1i}, \hat{\sigma}_{1i})$ be the Gaussian density functions for variable $i$ for class 0 and 1, respectively, with estimates of the mean (the sample mean) and standard deviation (sample standard deviation) plugged in. We will use $x$ to denote a vector of 300 observed variable values and $\mu_0, \mu_1$ and $\sigma_0, \sigma_1$ to denote the vectors of the corresponding means and standard deviations for classes 0 and 1. Denote by $I$ the indicies of the variables (ranging $0, 1, ..., 299$) retained as informative by the LASSO with $\lambda_{1se}$. Then, if all variables corresponding to indicies $I$ do, in fact, have different class conditional Gaussian distributions, an estimate of the optimal feature is $\frac{\prod_{i \in I} f(x_i|\hat{\mu}_{1i}, \hat{\sigma}_{1i})}{\prod_{i \in I} f(x_i|\hat{\mu}_{0i}, \hat{\sigma}_{0i})}$. However, supposing that we use a logistic regression as our model to make final predictions, it makes sense to take the log of this feature. This is because

$$\log\left(\frac{P(\text{class} = 1|x)}{P(\text{class} = 0|x)}\right) = \log\left(\frac{f(x|\mu_1, \sigma_1)}{f(x|\mu_0, \sigma_0)}\right) + \log\left(\frac{\pi_1}{\pi_0}\right), \tag{1}$$

where $\pi_0$ and $\pi_1$ are the overall class probabilities for class 0 and 1. So, we see that the log-odds should be linear in the log-likelihood ratio.

Instead of using this single feature, however, we instead use $|I|$ individual log-likelihood ratios as features. If we use a logistic regression model, we see that using $|I|$ individual log-likelihood ratios as features reduces to the single feature case when all of the regression coefficients are the same. Thus, we allow for slightly more flexible models by using univariate likelihood ratios, but any model class using univariate likelihood ratios contains the single feature model as a submodel.

### 3.1.2 Kernel Density Likelihood Ratios

The kernel density approach, used as an alternative to Gaussian log-likelihood ratios, was calculated in the same manner as the Gaussian log-likelihood ratios, with the exception of how the density estimates were calculated conditional on each class. Kernel density estimates, using the `density` function in `R` were used to estimate the density for each class rather than an assumed Gaussian density with an estimated mean. While tests for normality bolstered support for assuming Gaussian distributions, the nature of the competition also led us to consider a data-based density estimate as an alternative.

However, this also creates the possibility of overfitting the likelihood ratio estimates to the small sample size given in the training data. In fact, correlations between Gaussian log-likelihood ratios and kernel density log-likelihood ratios were relatively low; the two methods produced rather different likelihood ratio estimates. This is most likely due to the kernel density estimates being too sensitive to small "bumps" in the class-conditional densities (see again Figure 3), whereas the Gaussian log-likelihood ratios were more robust to noise present in the features.

When kernel density log-likelihood ratios were used as the set of LR features instead of Gaussian log-likelihood ratio estimates, predictions submitted to the Kaggle public leaderboard were lower; these data-based density metrics are prone to the overfitting we are hoping to avoid.

### 3.2 Conclusions of Feature Engineering

Feature engineering concluded with the following features remaining:

- Column '33' - raw data
- Column '65' - raw data
- Gaussian log-likelihood ratios for column indices in $I$.

## 4 Prediction Methods

With a final set of features, we can try multiple methods for final class predictions.

### 4.1 LASSO

Our second best performing model was fit using a LASSO. Features used in the model included all of the raw columns as variables plus the log-likelihood ratio features of the variables with column indicies in $I$. The penalty parameter used was the value of $\lambda_{1se} \approx 0.103$ computed by the `cv.glmnet` function in the `glmnet` package. This model ended up only including variables 33 and 65 and had coefficients $\hat{\beta}_{33} \approx 0.329$ and $\hat{\beta}_{65} \approx 0.149$. The intercept was estimated to be $\hat{\beta}_0 \approx 0.638$. Interestingly, fitting a logistic regression using only variables 33 and 65 results in substantially worse accuracy indicating that shrinking these regression coefficients is critical.

### 4.2 Random Forest

We also tried several implementations of random forest. Random forest, using the `randomForest` package in concert with the `caret` package, performed poorly with default parameter values. However, random forest predictions improved to an AUROC of 0.780 on the public leaderboard and 0.773 on the private leaderboard when fit using out-of-bag error and assigning prior class weights. This is a far cry from our best AUROC score of 0.833 mentioned in the LASSO section; however, it shows significant improvement over the random forest scores using default methods, which often scored in the $(0.40, 0.50)$ range. We saw these low scores firsthand through our own submissions as well as discussions on the Kaggle competition page. However, it does demonstrate that random forest requires careful parameter training and specification but can do well if fit carefully.

### 4.3 Combining Predictors

Our second high-scoring submission, aside from the LASSO discussed above, was a combination of previously submitted predictors from a variety of methods. A set of predictions from the LASSO model using Gaussian log-likelihood ratios, a set of predictions from a random forest fit with OOB error, and finally, a set of predictions from the LASSO model fit using kernel density log-likelihood ratios, were combined by averaging the predicted values. Since our best performing model prior to combining predictors was the LASSO with Gaussian log-likelihood ratios, we chose to combine that along with predictions from a kernel density log-likelihood to investigate whether striking a balance between the more robust Gaussian LRs and the more flexible kernel LRs would provide better results. In addition, predictions from the random forest using Gaussian LRs mixed in a differing prediction method. Each set of predictions used had correlations with each other that were at or below .85. This combined set of predictors resulted in the highest AUROC score on the public leaderboard, of 0.848, but tied with the best LASSO predictions on the private leaderboard, with a score of 0.833.

This method was chosen for combining predictions, rather than a true "ensemble" model, due to the fact that we had multiple different sets of features that generated some of our best sets of predictors. We felt this might be a good way to strike the balance between flexible and robust predictors.

## 5 Discussion

Our two final solutions both scored 0.833, which was 0.015 lower than our score on the public data set. We also dropped 147 places. Interestingly, not many teams changed more than $O(10)$ places, and the top scoring solution on the public data was also the top scoring solution on the private data. As we have discussed in class, it's possible that the winner simply correctly guessed a model that was close to the model that generated the data. This would make sense as the winner had around 350 submissions. However, the winning accuracy dropped something around 0.044, which was almost a three-fold drop in accuracy over our solutions. In fact, it seems like several of the highest scoring models suffered larger performance drops than some of the lower scoring teams.

It is interesting to note that our combined predictions did not score any higher on the private leaderboard than our best single set of predictions. However, since this method was not true "stacking" or "ensembling", but rather prediction averaging, it is not surprising that the model did not increase our ability to separate data into two classes. In this particular competition, it was very difficult to stick to a traditional statistical learning paradigm of first engineering one set of features, then using a wide variety of prediction methods. While this would have been ideal, the ambiguity and apparent randomness of the data led to more of a "feedback loop" between feature engineering and prediction methods than initially anticipated. In some ways, submitted predictions were needed to determine whether features were capturing any signal in the data. This is particularly true due to the small sample size of the training data.

One frustrating component of this competition was how little the knowledge of "optimal" features seemed to help over doing something completely naive (i.e. LASSO using raw data). The data were very likely simulated, and we attempted to take advantage of that strong suspicion by making (mostly) testable and

reasonable assumptions about the class conditional distributions. Yet, at the end of the day, one of our most competitive models was essentially just a logit-linear model with two raw data variables which seemed to have different means between the classes. The theoretical justification for log-likelihood ratios being the transformation that should be linearly related to the logit posterior class probabilities seemed to be contradicted by model performance (all log-likelihood ratios were kicked out of the LASSO). With that being said, it is, of course, possible that all of the predictor variables were just generated iid $N(0,1)$, and then the data creator plugged those variables into the oracle model (which must be very similar to a simple logistic regression). In this case, the class conditional densities need not be multivariate normal. Further, the optimal features would then just simply be the untransformed predictor variables. Perhaps it was for this reason that the likelihood ratio features were all ejected from our best LASSO model. Whether or not this is true, however, the takeaway seems to be that it is easy to be fooled into making incorrect assumptions about the data generating mechanism even when those assumptions largely pass statistical scrutiny with flying colors.

With reference to the above discussion, one thing that would have been interesting would have been to try to identify the class conditional distributions if the data were simulated in the hypothesized manner. That would have potentially provided some indication of how easy it would be to get fooled into thinking that the class conditional distributions are Gaussian.

One might take issue with the fact that while most of our assumptions about class conditional distributions were tested in a reasonable way, our assumption of independence and/or zero correlation between columns was not ideal. We have discussed how the class conditional data could be multivariate normal with zero correlation, or be marginally Gaussian with strange dependencies between variables. One attempt of dealing with the latter was to take the subset of predictor columns corresponding to indices $I$ and use more flexible methods, such as a random forest, boosted trees, single classification trees, Quinlan's C5.0. The idea here was that more flexible predictors might be able to account for interactions or dependence between the whatever variables that, by themselves, seemed to be important. The performance of these methods never matched that of our best performing model, but using these methods/algorithms on the subset of the data worked better than using the methods on the entire data set.

At the end of the day, this competition was an interesting learning experience for our team and gave us practice with investigating data with no added context. It forced us to think about the mathematical concepts behind the prediction methods more than a traditional contest may have. Since we were essentially flying blind on context information, carefully navigating which assumptions we can and should make was a good exercise.

# 6 Appendices

See attached.